

**Amendments to the Claims:**

This listing of claims will replace all prior versions, and listings, of claims in the application:

**Listing of Claims:**

1. (Currently amended) A method for protecting data as it passes through a buffering device that connects protocols that use different block sizes or unblocked data, said method comprising:

allocating a plurality of blocks of storage in a first memory in said buffering device for storing a transfer length of unblocked data;

receiving said data on a first port from a source, said source determining said transfer length;

writing storing said data, in a first memory in said buffering device as said data is received, to successive ones of said plurality of blocks until an end of said transfer length is reached such that said data is stored in a plurality of blocks, each one of said plurality of blocks  $[[block]]$  having a length of  $2^n$ , where n is a positive integer;

if an end one of said plurality of blocks that includes an end of said transfer length is not full of data, adding padding to said end one of said plurality of blocks until said end of said plurality of blocks is complete, wherein padding is added only to an end one of said plurality of blocks that includes an end of said transfer length until the end one of the plurality of blocks is complete;

as said data is being written to each one of said plurality of blocks, calculating a running  $[[first]]$  cyclical redundancy code for each of said plurality of blocks as each of said plurality of blocks is written;

when writing to each one a first block of said plurality of blocks is completed, storing, for each one of said plurality of blocks, a final value of said running cyclical redundancy code that was calculated for each one of said plurality of blocks as a corresponding first cyclical redundancy code in a second memory on said buffering device; and

when writing said data to a second port, computing a second cyclical redundancy code for each of said blocks of said plurality of blocks and if said second cyclical redundancy code corresponding to a given block is equal to said first cyclical redundancy code corresponding to said given block, writing said given block to said second port.

2. (Original) The method of Claim 1, wherein the data is received with a protection code that is checked and discarded.

3. (Original) The method of Claim 1, wherein said buffering device is a DDR device connected between a bus and a tape drive.
4. (Original) The method of Claim 1, wherein one of said first and said second ports is connected to a protocol that does not use fixed block lengths.
5. (Original) The method of Claim 1, wherein locations in said second memory are mapped to locations in said first memory.
6. (Canceled)
7. (Currently amended) A device for buffering data between two protocols, at least one of which does not utilize blocks, said device comprising:
  - a first port connected to communicate using a first protocol of said two protocols;
  - a second port connected to communicate using a second protocol of said two protocols;
  - a cyclical redundancy code engine connected to be selectively connected to one of said first port and said second port;
  - a plurality of blocks of storage that have been allocated for storing a transfer length of unblocked data;
  - a first random access memory connected to said cyclical redundancy code engine and including said plurality of blocks for storing said data that is in which data passing between said first port and said second port is stored in fixed size blocks, said data being written to successive ones of said plurality of blocks as said data is received until an end one of said transfer length is reached, each one of said plurality of blocks having a fixed size;
  - if an end one of said plurality of blocks that includes an end of said transfer length is not full of data, padding being added to said end one of said plurality of blocks until said end of said plurality of blocks is complete, wherein padding is added only to an end one of said plurality of blocks that includes an end of said transfer length;
  - said cyclical redundancy code engine calculating a running cyclical redundancy code for each one of said plurality of blocks as data is written to each one of said plurality of blocks;

a second random access memory connected to said cyclical redundancy code engine for storing, for each one of said plurality of blocks, a final value of said running cyclical redundancy code that was calculated for each one of said plurality of blocks as a first cyclical redundancy code when writing to each one of said plurality of blocks is completed and in which first cyclical redundancy codes corresponding to said fixed size blocks are stored; and

a comparator connected to compare a second cyclical redundancy code calculated for each one of said plurality of blocks as said fixed size blocks are written with said first cyclical redundancy code calculated when writing to each one of said plurality of blocks is completed ~~said fixed size blocks were written;~~

whereby the data passed through said device is protected by a cyclical redundancy code.

8. (Original) The device of Claim 7, wherein said cyclical redundancy codes are stored in said second random access memory in a mapped relationship to said fixed size blocks stored in said first random access memory.

9. (Original) The device of Claim 7, further comprising a protection module connected to said first port for checking a protection code that is received and discarding said protection code.

10. (Original) The device of Claim 7, wherein locations in said second random access memory are mapped to locations in said first random access memory.

11. (Canceled)

12. (Currently amended) A computer program product on a computer-readable device, comprising the computer implemented steps of:

first instructions for allocating a plurality of blocks of storage for storing a transfer length of unblocked data;

second [[first]] instructions for receiving said data on a first port from a source, said source determining said transfer length;

third ~~second~~ instructions for writing storing said data to [[in]] a first memory in said buffering device, as said data is received, to successive ones of said plurality of blocks until an end of said transfer

length is reached such that said data is stored in a plurality of blocks, each one of said plurality of blocks  
[[block]] having a length of  $2^n$ , where n is a positive integer,

if an end one of said plurality of blocks that includes an end of said transfer length is not full of  
data, fourth instructions for adding padding to said end one of said plurality of blocks until said end one  
of said plurality of blocks is complete, wherein padding is added only to an end one of said plurality of  
blocks that includes an end of said transfer length;

fifth [[third]] instructions for calculating a running first cyclical redundancy code for each of said  
plurality of blocks as data is being written to each one of said plurality of blocks each of said plurality of  
blocks is written;

sixth fourth instructions for storing, when writing to each one a first block of said plurality of  
blocks is completed, storing, for each one of said plurality of blocks, a final value of said running cyclical  
redundancy code that was calculated for each one of said plurality of blocks as a corresponding first  
cyclical redundancy code in a second memory on said buffering device; and

seventh [[fifth]] instructions for computing, when writing said data to a second port, a second  
cyclical redundancy code for each of said blocks of said plurality of blocks and if said second cyclical  
redundancy code corresponding to a given block is equal to said first cyclical redundancy code  
corresponding to said given block, writing said given block to said second port.

13. (Currently amended) The computer program product of Claim 12, wherein said second [[first]]  
instructions check a protection code received with the data and discarded said protection code.

14. (Original) The computer program product of Claim 12, wherein said computer program product is  
embodied on a protocol interface device connected between a bus and a tape drive.